Jussi Laakkonen

Kolmiulotteinen pelisovellus taulutietokoneelle

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinöörityö

29.4.2013

Metropolia

Tekijä Otsikko Sivumäärä Aika	Jussi Laakkonen Kolmiulotteinen pelisovellus taulutietokoneelle 38 sivua 29.4.2013
Tutkinto	insinööri (AMK)
Koulutusohjelma	mediatekniikka
Suuntautumisvaihtoehto	digitaalinen media
Ohjaajat	yliopettaja Harri Airaksinen lehtori Peter Hjort

Insinöörityön tavoitteena oli suunnitella ja toteuttaa kolmiulotteinen pelisovellus taulutietokoneelle.

Suunnitteluvaiheeseen sisällytettiin taustatutkimusta mobiilipelisovellusten liiketoiminnasta ja ansaintamalleista. Mobiilipelien myynti eroaa konsolipelien myynnistä siinä, että se on keskittynyt digitaaliseen jakeluun. Nopeutuvien mobiililaajakaistayhteyksien ansiosta pelikauppa on aina kuluttajan mukana ja ostotapahtuman jälkeen peli on pelattavissa hetkessä. Pelisovellusten sisällä suoritettaviin mikromaksuihin perustuvan ansaintamallin liikevaihto on ohittanut vanhan kappalemääräiseen myyntiin perustuvan mallin. Taloudellisesti menestyneimpien mobiilipelien perusversiot jaetaan käyttäjille lähes poikkeuksetta ilmaiseksi. Liiketoimintamallien muuttuessa myös pelinkehittäjien toimintatavat ovat muuttumassa.

Pelisovelluksen teknisen toteutuksen suunnittelussa tutkittiin kolmea eri pelinkehityksen teknologiaa, joista yksi valittiin sovelluskehityksen pääteknologiaksi. Päätökseen vaikuttavia asioita olivat teknologian hinta, ominaisuudet ja pelisovelluksen tekniset vaatimukset. Parhaaksi vaihtoehdoksi todettiin Objective-C-ohjelmointikielellä kirjoitettu NinevehGL, joka on ilmainen kehitysvaiheessa oleva 3D-grafiikan piirtämiseen tarkoitettu kirjasto. Myös muita kehityksessä käytettäviä teknologioita ja työkaluja valittaessa pyrittiin suosimaan ilmaisia ja avoimen lähdekoodin vaihtoehtoja.

Insinöörityön lopputuloksena saatiin aikaiseksi ulkoavaruuteen sijoittuva Bounceroids-pelisovellus. Sovelluksen kaikki sisältö luotiin yksin, mikä luonnollisesti rajoitti työn laajuutta ja teknisiä ratkaisuja. Toteutuksessa hyödynnettiin eri mediatekniikan osa-alueita sisällöntuotannosta, ohjelmistokehityksestä, graafisesta suunnittelusta, 3D-mallinnuksesta ja äänituotannosta. Peli julkaistaan sovelluskaupassa toukokuussa 2013.

Avainsanat	iOS, iPad, NinevehGL, peliohjelmointi, 3D-grafiikka

Author	Jussi Laakkonen
Title	3D game application for tablet computers
Number of Pages	38 pages
Date	29 April 2013
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructors	Harri Airaksinen, Principal Lecturer Peter Hjort, Senior Lecturer

The main purpose of this thesis was to design and implement a 3D game application for tablet computers.

The design phase included background research into common business models used by the mobile gaming industry. What differentiates the sale of mobile games from console games is the total reliance on digital distribution. The actual game store has been brought straight into the users pocket due to the ever increasing speeds of mobile broadband connections. After a game is purchased it is downloadable and playable within minutes. Revenues from micro transactions inside mobile games have surpassed the traditional business models where revenues are based on the sale of applications. The most financially successful mobile games are all given free of charge to the end user. As business models are changing so are the ways mobile games are being developed.

Three iOS game development technologies were researched and compared while designing the technical implementation of this thesis. One of them was chosen as the main technology for the game development. The most important things that were considered when choosing the technology were the cost, features and the technical requirements for the game. The best choice was deemed to be NinevehGL which is a free development phase library written in the Objective-C programming language. Open source and free technologies were generally favored when selecting tools for this thesis.

As a result the outer space themed Bounceroids iPad game was made. All the game's content was made by one person and the scope and technical choices were limited to the developer's skillset. Various media technology related skills were used in the development of the game including software engineering, graphic design, 3D modeling and sound production. The game will be released in the App Store in May 2013.

Keywords	iOS, iPad, NinevehGL, game programming, 3D graphics

Sisällys

Lyhenteet

ARC	Automatic reference counting. Objective-C-ohjelmointikielen kääntämisen yhteydessä suoritettava automaattinen muistinhallinta.
ARM	Advanced RISC Machines. Mobiililaitteissa yleisesti käytössä oleva 32-bittinen prosessoriarkkitehtuuri.
eCPM	Effective cost per mille. Hinta, jonka mainostaja maksaa, kun mainos näytetään tuhannelle ihmiselle.
FSAA	Full screen anti-aliasing. Digitaalinen signaalinkäsittelymenetelmä, jolla voidaan poistaa teräviä reunoja 3D-mallien ympäriltä.
GLSL	OpenGL Shading Language. C-kielen syntaksilla kirjoitettava korkean tason ohjelmointikieli OpenGL-rajapintaan.
IDC	International Data Corporation. Kansainvälinen tieto- ja viestintäteknologian markkinatutkimukseen keskittyvä yritys.
KISS	Keep it simple, stupid. Suunnitteluperiaate, jonka mukaan järjestelmät toimivat parhaiten, kun ne suunnitellaan yksinkertaisiksi.
MVC	Model-view-controller. Ohjelmistoarkkitehtuurityyli, jossa käyttöliittymä erotetaan tietomalleista.
SDK	Software development kit. Kokoelma ohjelmistokehitystyökaluja, jotka mahdollistavat omien sovelluksien luomisen tietylle käyttöjärjestelmälle tai laitteelle.
WWDC	Apple Worldwide Developers Conference. Ohjelmistokehittäjille vuosittain järjestettävä konferenssi, jossa Apple esittelee uusia teknologioita ja työkaluja.

1 Johdanto

Insinöörityön tavoitteena on suunnitella ja toteuttaa pelisovellus Applen iPad-taulutietokoneelle. Motivaationa työn tekemiselle on tarve saada yleinen käsitys erilaisista iOS-pelinkehittäjille tarjolla olevista teknologioista. Toteutuksessa pyritään erityisesti hyödyntämään ilmaisia ja avoimen lähdekoodin teknologioita kaupallisten vaihtoehtojen sijasta. Ilmaisten teknologioiden käytön päätarkoituksena on luoda julkaisuvalmis tuote minimoimalla taloudelliset riskit ja kustannukset.

Työssä perehdytään mobiilipelien nykyiseen markkinatilanteeseen, suurimpiin myyntikanaviin ja yleisimpiin ansaintamalleihin. Myyntikanavista tutkimuksen kohteina ovat suuren markkinaosuuden saavuttaneet App Store- ja Google Play -sovelluskaupat. Osion tarkoituksena on saada yleiskuva jännittävästä ja nopeasti muuttuvasta mobiilipeliteollisuudesta, jonka ennustetaan kasvavan voimakkaasti tulevina vuosina.

Lisäksi tarkastellaan kahta yleisesti iOS-pelinkehityksessä käytössä olevaa teknologiaa, jotka ovat keränneet suuren kehittäjäyhteisön. Unity-ohjelmistoa ja Cocos2d-ohjelmistokehystä vertaillaan insinöörityössä käytettävään varsin nuoreen NinevehGL-kirjastoon. Tarkoituksena on selvittää, onko NinevehGL varteenotettava työkalu pelisovelluksen luomiseen. NinevehGL on vielä vahvasti kehitysasteella, ja kaikki siihen liittyvä dokumentaatio ja tieto löytyy lähes yksinoikeudella kirjastoa kehittävän DB-Interactivelyn verkkosivuilta. Tärkein tieto aiheesta löytyy toistaiseksi kirjaston luokkakuvauksista.

Viimeisessä osassa käsitellään insinöörityötä varten toteutettavan Bounceroids-pelisovelluksen suunnittelua, sisällöntuotantoa ja ohjelmistokehitysmenetelmiä. Osioon sisältyvät kaikki pelisovelluksen luontiin liittyvät vaiheet konseptista, käyttöliittymäsuunnittelusta toteutukseen ja 3D-grafiikan sekä äänien tuottamiseen. Kaikki sisällöntuottamisessa käytettävät ohjelmistot ja niiden ominaisuudet esitellään pintapuolisesti. Teknisten ratkaisujen esittelyssä tullaan painottumaan ohjelmointiin, joka tehdään Objective-C-ohjelmointikielellä hyödyntäen iOS-käyttöjärjestelmän ja kolmannen osapuolen ohjelmistokehyksiä. Aiheen sisäistäminen vaatii lukijalta ainakin korkean tason 3D-

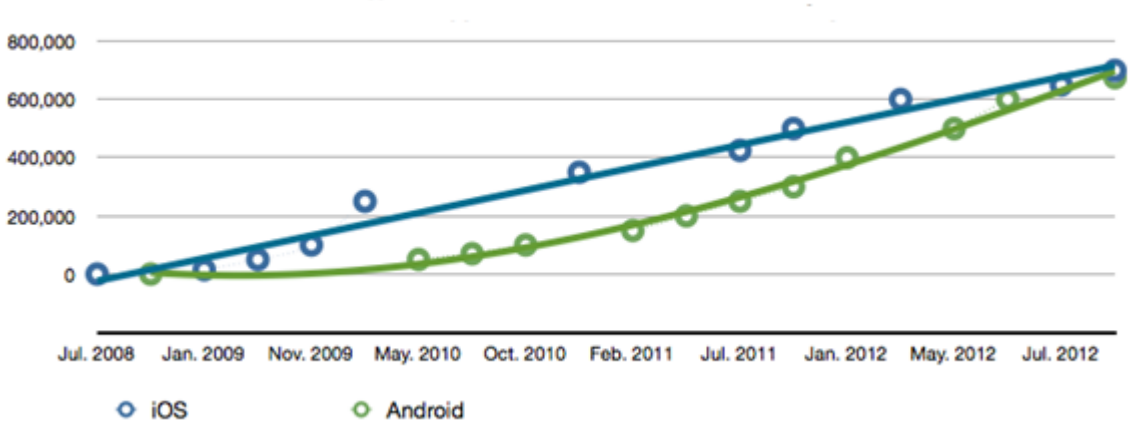
grafiikan ja ohjelmoinnin tuntemusta, koska insinöörityön tarkoituksena ei ole käydä läpi peliohjelmonnin perusteita.

2 Mobiilipelit liiketoimintana

2.1 Markkinat

Älypuhelimien yleistymisen myötä mobiilipelit ovat kehittyneet yksinkertaisista laitevalmistajien luomista matopeleistä miljardien dollareiden arvoiseksi liiketoiminnaksi. Mobiilipelimarkkinoiden arvoksi arvioitiin viime vuonna 7,8 miljardia dollaria, ja markkinoiden arvon odotetaan kasvavan 18 miljardiin dollariin vuoteen 2016 mennessä [1]. Yksi markkinoiden kasvun avaintekijöistä on älypuhelimien myynti. Kovan kilpailun ja tekniikan kehittymisen myötä älypuhelinten hinnat laskevat ja mahdollisten asiakkaiden määrä kasvaa vuosi vuodelta. ComScoren tutkimuksen mukaan jopa 73 % Euroopassa myydyistä puhelimista on älypuhelimia [2]. Markkinatutkimusyhtiö IDC:n (International Data Corporation) arvion mukaan viime vuonna älypuhelimia myytiin maailmanlaajuisesti yli 700 miljoonaa kappaletta. Kasvua edelliseen vuoteen syntyi 44 %. [3.]

Noin 92 % vuoden 2012 viimeisellä neljänneksellä myydyistä älypuhelimista sisälsi joko Android- tai iOS-käyttöjärjestelmän [5]. Vastaavasti kaksi suurinta mobiilipelien myyntikanavaa ovat Androidin Google Play ja iOS:n App Store -sovelluskaupat. Google Play ohitti App Storen sovelluksien määrässä vuoden 2012 syksyllä, jolloin molemmissa kaupoissa oli noin 700 000 sovellusta [4]. Kuvassa 1 näkyvät Googlen ja Applen julkaisemat luvut sovelluskauppojen kasvusta vuosina 2008–2012. Suuren markkinaosuuden hallitsevassa asemassa sovelluskauppojen ylläpitäjät Apple ja Google pystyvät ottamaan 30 % sovelluskaupan myyntituotoista itselleen. IDC:n ennusteiden mukaan Microsoftin Windows Phone -käyttöjärjestelmän markkinaosuus kasvaa yli 11 prosenttiin vuoteen 2016 mennessä, mikä saattaa lisätä kilpailua sovelluskauppojen hinnoittelumalleissa [6].



Kuva 1. App Store- ja Google Play -sovelluskauppojen sovellusten määrä vuosina 2008–2012 [4].

Alkuvuonna 2010 julkaistusta iPad-taulutietokoneesta muodostui myyntimenestys, joka loi älypuhelimien rinnalle uudenlaisen tuotekategorian. Menestyksen myötä useat eri laitevalmistajat ovat alkaneet myydä älypuhelinten lisäksi taulutietokoneita. Kolme suurinta valmistajaa ovat Apple, Samsung ja Amazon [7]. Taulutietokoneita myytiin vuonna 2012 IDC:n mukaan lähes 120 miljoonaa kappaletta [8]. Myynnin arvioidaan kaksinkertaistuvan vuoteen 2016 mennessä. Taulutietokoneilla on merkittävä rooli mobiilipelien tulevaisuudessa, koska 67 % niiden käyttöajasta käytetään pelaamiseen [9]. Älypuhelimilla pelaamiseen käytettävä aika on vain 39 %. Tämän lisäksi käyttäjät viettävät taulutietokoneen kanssa keskimäärin enemmän aikaa käyttökertaa kohden. Taulutietokoneiden käyttöön kuluvat pidemmät yhtäjaksoiset käyttöajat, niiden suuremmat näytöt ja älypuhelimia tehokkaammat prosessorit antavat pelinkehittäjille mahdollisuuden luoda teknisesti ja sisällöllisesti rikkaampia tuotteita.

Suuret PC- ja konsolipelejä julkaisevat yritykset, kuten Electronic Arts, Vivendi ja Namco Bandai, ovat kukin laajentaneet tuotevalikoimaansa mobiilipeleihin [10]. Pelijulkaisijan näkökulmasta mobiilipelien edut konsolipeleihin verrattuna ovat alhaisemmat kehityskustannukset, lyhyemmät kehitysaikataulut ja sovelluskauppojen tarjoama valmis jakelualusta [11]. Digitaalinen jakelu on haastamassa konsolipelien nykyisen jakelumallin, joka perustuu jälleenmyyjille toimitettavien kappaleiden myyntiin. Google on madaltanut kynnystä sovelluskehittäjille niin alas, että 25 dollarin kertaluontoisella lisenssimaksulla kuka tahansa voi luoda ja julkaista mobiilipelin miljoonille mahdollisille asiakkaille [12].

2.2 Ansaintamallit

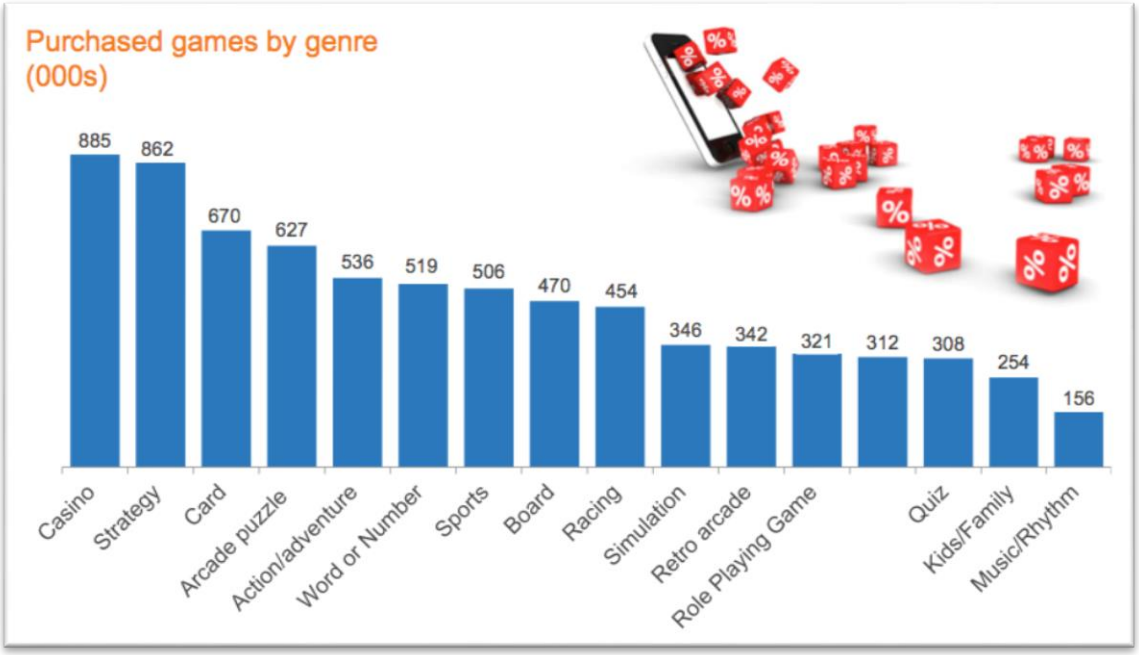
Mobiilipelit hyödyntävät useita eri ansaintamalleja, eikä konsolipeleille tyypillinen pelien kappalemääräiseen myyntiin perustuva malli ole aina tuottavin. Apple ylläpitää listaa eniten tuloja keränneistä mobiilisovelluksista. Vuoden 2012 iPhone-sovellusten listan kymmenen kärkisijaa saavutti kymmenen peliä, joille yhteistä on niiden käyttämä free-to-play-ansaintamalli [13]. Tässä ansaintamallissa pelisovellus annetaan käyttäjille ilmaiseksi ja julkaisijan saamat tulot muodostuvat sovelluksen sisällä tehdyistä mikromaksuista. Tyypillisesti mikromaksuilla ostetaan peliin lisäsisältöä tai pelin kulkua muuttavia ominaisuuksia. Mikromaksut veloitetaan suoraan sovelluskauppaan ilmoitetulta luottokortilta, minkä jälkeen ostettu tuote on välittömästi pelaajan käytettävissä. Esimerkiksi listan 8. sijan Clash of Clans -pelissä pelaaja voi ostaa jalokiviä, jotka nopeuttavat pelaajan edistymistä. Kuvassa 2 on Clash of Clans -pelin sisäinen jalokivikauppa.

Kuva 2. Clash of Clans -pelin sisäinen jalokivikauppa.

Mikromaksujen lisäksi ilmaiset mobiilipelit voivat tuottaa tuloja sovelluksen sisäisistä mainoksista. Sekä Google että Apple tarjoavat omat ohjelmistokehyksensä mainoksien

upottamiseen sovelluksiin. Vuonna 2009 Googlen omistukseen siirtynyt AdMob on yksi maailman suurimmista mobiilimainoksia tarjoavista palveluista. Vuoden 2011 lopussa AdMob-mainoksilla oli Google Play -sovelluskaupassa lähes 90 %:n markkinaosuus [14]. Mobiilisovelluksissa näytettävien mainosten hinnoittelu on huomattavasti verkkosivustojen mainoksia alhaisempi. Lokakuussa 2012 julkaistun tutkimuksen mukaan mobiilisovellusten mainoksien eCPM (effective cost per mille) -hinta on vain 0,75 dollaria, kun verkkosivustoilla hinta on 3,50 dollaria [15]. Yhdeksi alhaisten hintojen syyksi epäillään suurien mainostajien hitautta mobiilimainontaan siirtymisessä. Alhaisten hintojen myötä näytettävien mainosten lukumäärään perustuvan ansaintamallin rinnalle on syntynyt sovelluksien asennuksiin perustuva vaihtoehto, jonka tuotot ovat tavallisia mainoksia korkeammat [16]. Tässä ristimarkkinoinnin mallissa mainoksesta maksetaan vasta siinä vaiheessa, kun mainoksen avannut henkilö asentaa laitteeseensa mainoksessa mainostettavan sovelluksen.

Sovelluksia voi myös julkaista ilmaiseksi ilman mainoksia tai minkäänlaisia mikromaksuja. Ilmaiset sovellukset ovat paljon maksullisia suositumpia. ICT-alan tutkimusyritys Gartner arvioi viime vuonna 89 %:n kaikista sovelluskaupasta ladatuista sovelluksista olevan ilmaisia [17]. Kuvassa 3 ovat ComScoren tutkimuksen perusteella suosituimmat pelisovellusten lajityypit. Listan kärjessä ovat kasino-, strategia- ja korttipelit.



Kuva 3. Suosituimmat pelisovellusten lajityypit [2].

3 iOS-pelinkehitys

3.1 Työkalut ja teknologiat

Insinöörityössä vertailtiin kolmea erilaista iOS-pelinkehityksessä hyödynnettävää teknologiaa, joista kaksi on suuren käyttäjämäärän keränneet Cocos2d- ja Unity ja kolmas on tämän opinnäytetyön Bounceroids-mobiilipelin kehitykseen valittu NinevehGL-kirjasto. Näiden lisäksi iOS-pelinkehittäjille on tarjolla laaja valikoima sekä ilmaisia avoimen lähdekoodin että kaupallisia suljetun lähdekoodin työkaluja, pelimoottoreita ja kirjastoja. Pelinkehitystyökalujen tarkoitus on helpottaa ja nopeuttaa pelien luomisprosessia esimerkiksi graafisen käyttöliittymän avulla. Pelimoottorit abstrahoivat pelisovelluksille yleiset tietokonegrafiikan piirtämiseen, verkkoliikenteeseen, fysiikan laskentaan, ääneen tai tekoälyyn liittyvät rutiinit ja antavat ohjelmoijalle korkeamman tason rajapinnan niiden suorittamiseen ilman käyttöjärjestelmän rajapintojen tuntemusta. Muita pelinkehityksessä hyödyllisiä kirjastoja on esimerkiksi muistinhallintaan ja matemaattisten rutiinien suorittamiseen.

3.2 Unity-pelimoottori

Unity on Unity Technologies -yrityksen kehittämä järjestelmäriippumattomaksi tarkoitettu kaupallinen pelimoottori, joka sisältää myös integroidun kehitysympäristön. Ensimmäinen versio julkaistiin Applen WWDC-konferenssissa vuonna 2005, jolloin se mahdollisti pelinkehityksen vain Mac OS X -käyttöjärjestelmälle. Nykyään pelimoottori tukee kymmentä eri alustaa, joista kaksi merkittävintä mobiilikäyttöjärjestelmää ovat Android ja iOS. [18.]

Unity tarjoaa pelinkehittäjille erilaisia lisenssivaihtoehtoja, joista pienyrityksille ja harrastelijoille suunnattu pelkistetty versio on ilmainen. Ammattimaiseen iOS-pelinkehitykseen tarkoitetut Unity Pro- ja iOS Pro -lisenssit maksavat yhteensä 3 000 dollaria. Unity on Game Developer -lehden tutkimuksen mukaan suosituin mobiilipelien kehityksessä käytettävä pelimoottori 53,1 %:n markkinaosuudella, ja sitä on käytetty yli 1 500:ssa App Storessa julkaistussa mobiilipelissä mukaan lukien suomalaisissa Bad Piggies- ja Rochard-peleissä. Lisensoituja pelinkehittäjiä Unitylla on yli 1,2 miljoonaa. [18.]

Unity sisältää projektinhallintatyökalun, jossa on graafinen käyttöliittymä ja mahdollisuus reaaliaikaisesti esikatsella pelimoottorin piirtämää 3D-maisemaa. Graafisen käyttöliittymän ansiosta pelinkehitystä voidaan tehdä myös ilman ohjelmointikielien tuntemusta. Kehitysympäristöön on mahdollista tuoda kaikkia yleisimpiä 3D-mallintamiseen tarkoitettujen ohjelmistojen tuottamia tiedostomuotoja, piirto-ohjelmien kuvaformaatteja ja video- ja äänitiedostoja ilman erillistä manuaalista konversiota. Sen lisäksi pelinkehittäjät voivat ostaa muiden käyttäjien luomia 3D-malleja, kuvia, ääniä, koodin pätkiä ja esimerkkiprojekteja Unity Asset Store -verkkokaupasta, joka on integroitu mukaan ohjelmistoon. Unity iOS Pro -lisenssi tukee OpenGL ES 2.0 -standardia, shader-ohjelmia, fysiikkalaskentaa, kolmiulotteisia ääniä, partikkeleita, reaaliaikaisia varjoja ja kuvien jälkikäsittelyä. Unityssa ohjelmointi voidaan tehdä UnityScript-, C#- tai Boo-ohjelmointikielellä, ja se tehdään yleensä ohjelmiston mukaan sisällytetyssä MonoDevelop-sovelluksessa. [19.]

Unity on ominaisuuksiltaan erittäin laaja. Se on uusimman sukupolven kaupallisten pelimoottoreiden kärkeä, ja siitä on vaikea löytää teknisiä puutteita vertaamalla sitä muihin iOS-pelinkehityksessä käytettäviin pelimoottoreihin tai ohjelmistokehyksiin. Kuten muissa suljetun lähdekoodin ohjelmistoissa, kehittäjälisenssin ostaneet asiakkaat ovat mahdollisia ongelmatilanteita ja virheitä kohdatessaan pelimoottoria kehittävän yrityksen aikataulujen armoilla. Tuhansien eurojen lisenssimaksut myös tarkoittavat sitä, ettei se sovi harrastelijoiden tai amatöörien projekteihin. Lisenssimaksujen suuruuden takia Unitya ei valittu insinöörityössä toteutetun pelin pääteknologiaksi. Suuren budjetin projekteihin Unity vaikuttaa selvästi parhaalta vaihtoehdolta.

3.3 Cocos2d-ohjelmistokehys

Cocos2d on alun perin Python-ohjelmointikielellä kirjoitettu kaksiulotteisten pelien kehittämiseen ja grafiikan piirtämiseen tarkoitettu ilmainen ohjelmistokehys. Se julkaistiin helmikuussa 2008, ja ensimmäinen Objective-C-ohjelmointikielelle käännetty versio julkaistiin saman vuoden kesäkuussa. Cocos2d on vuosien myötä kasvattanut suosiotaan, ja nykyään se on seitsemälle eri kielelle kirjoitettu avoimen lähdekoodin projekti. Objective-C-kielen versio eli Cocos2d for iPhone on kerännyt ympärilleen laajan kehittäjäyhteisön, joka luonut ja julkaissut yli 2 500 peliä App Storessa. [20.]

Cocos2d sisältää useita pelimoottoreille tyypillisiä ominaisuuksia, kuten kuvien jälkikäsittelyn, partikkelit, TMX-muotoiset kartat, CocosDenshion-äänikirjaston ja integroidun fysiikanlaskennan Box2d- ja Chipmunk-fysiikkamoottoreilla. Cocos2d-sovellukset ohjelmoidaan Objective-C-kielellä kirjoitettujen rajapintojen avulla. Kirjasto sisältää täyden dokumentaation, tutoriaaleja ja esimerkkiprojekteja. Asennuksen yhteydessä Cocos2d lisää omat kehysprojektinsa Xcoden projektilistaan, mikä helpottaa ohjelmistokehyksen integrointia sovelluksiin. [20.]

Koska kyseessä on kaksiulotteisten pelien kehittämiseen tarkoitettu teknologia, Cocos2d ei sisällä valmiita rajapintoja 3D-grafiikan piirtämiseen. Tämän vuoksi Cocos2d:n grafiikkarajapinnat on suunniteltu vain Sprite-grafiikalla tehtyihin ratkaisuihin. Sprite-grafiikka toteutetaan kaksiulotteisilla bittikarttakuvilla. Näyttävän lopputuloksen saavuttaminen vaatii suuren määrän kuvankäsittelyä, ja kuvien animointi on työläämpää kuin 3D-avaruudessa, missä animaatio voidaan toteuttaa suoraan liikuttamalla tai pyörittämällä valmista kolmiulotteista kappaletta. Cocos2d:tä ei valittu insinöörityössä käytettäväksi teknologiaksi, koska työssä toteutettu peli haluttiin piirtää 3D-avaruudessa. 3D-mallien käyttämisen arvioitiin tuottavan sovellukseen Sprite-grafiikkaa enemmän lisäarvoa pienemmällä työmäärällä.

3.4 NinevehGL-kirjasto

NinevehGL on brasilialaisen DB-Interactively-yrityksen kehittämä ilmainen koodikirjasto 3D-grafiikan piirtämiseen Applen iOS-käyttöjärjestelmän sovelluksissa. Ensimmäinen julkinen kehitysversio julkaistiin heinäkuussa 2011. Kehitysversion käyttöönotto vaatii rekisteröitymisen beta-testaajaksi NinevehGL-verkkosivuilla. Kehitysversio sisältää kirjaston binääri- ja header-tiedostot sekä esimerkkiprojekteja. DB-Interactively kertoo tavoitteenaan olevan koko projektin lähdekoodien julkaisemisen vuoden 2013 loppuun mennessä. [21.]

Kuten Cocos2d, myös NinevehGL lisää itsensä asennuksen yhteydessä Xcoden projektilistaan. Kirjasto sisältää Objective-C-kielellä toteutetun korkean tason rajapinnan, jonka tarkoituksena on tarjota iOS-sovelluskehittäjälle helpommin lähestyttävä vaihtoehto 3D-grafiikan ohjelmointiin. Näytönohjaimen OpenGL-rajapinnat on abstrahoitu, ja

kirjasto muistuttaa sekä arkkitehtuuriltaan että dokumentaatioltaan hyvin paljon iOS-ohjelmoijille tuttua Cocoa Touch -ohjelmistokehystä. [22.]

NinevehGL optimoi 3D-mallien lukua laitteen kiintolevyltä konvertoimalla ne sovelluksen ensimmäisen avauksen yhteydessä NGL-tiedostomuotoon, jonka laite pystyy kehittäjien mukaan lukemaan kiintolevyltä noin sata kertaa nopeammin kuin yleisesti käytetyt OBJ ja DAE -tiedostomuodot. Jos 3D-mallin tiedosto muuttuu tai se korvataan, NinevehGL suorittaa konvertoinnin uudestaan. [22.]

Vuoden 2011 maaliskuussa julkaistun iPad 2:n jälkeen kaikissa Applen taulutietokoneissa on ollut moniydinprosessori, joka mahdollistaa usean samanaikaisen laskutoimituksen suorittamisen laitteen prosessorissa. NinevehGL on suunniteltu käyttämään useaa ydintä suorituskyvyn parantamiseksi. Kehittäjällä on oletuksena vähintään kahdeksan säiettä käytössään: yksi säie on 3D-grafiikan piirtämiseen, yksi avustava säie tiedon siirtämiseen näytönohjaimelle ja viisi säiettä, jotka lukevat tiedostoja laitteen kiintolevyltä ja luovat olioita keskusmuistiin. Näiden lisäksi on käyttöliittymäsäie, joka ottaa vastaan kosketusnäytön kosketuksia ja suorittaa sovellukseen ohjelmoitua logiikkaa. Käytännössä kahdeksaa säiettä ei pystytä hyödyntämään täydellisesti, koska uusimman sukupolven iOS-mobiililaitteissa on korkeintaan neljä prosessoriydintä. Säikeistäminen kuitenkin mahdollistaa usean operaation suorittamisen samanaikaisesti prosessorissa, mikä nopeuttaa sovelluksen toimintaa huomattavasti. [22.]

NinevehGL tukee OpenGL ES 2.0 -standardin shader-ohjelmia. Shader-ohjelmat voidaan kirjoittaa GLSL-ohjelmointikielellä, joko omiin tiedostoihinsa tai suoraan sovelluksen koodiin NSString-muuttujaan. Shader-ohjelmat mahdollistavat 3D-mallien pintaan piirrettävät erikoistehosteet, kuten läpinäkyvyyden, kiiltämisen ja heijastukset. Kirjaston sisältämä Global-rajapinta mahdollistaa sovelluksen laajuisten asetusten muuttamisen yhden rivin C-funktioilla OpenGL-rajapinnan GL-funktioiden tapaan. [22.]

Kaksi vuotta vanha NinevehGL on varsin nuori teknologia. Aiheesta ei ole julkaistu vielä kirjallisuutta, joten kirjastoon perehtyvän ohjelmoijan täytyy olla valmis lukemaan dokumentaatiota ja tutkimaan kirjaston mukana tulevia esimerkkiprojekteja ja NinevehGL:n sivuilta löytyviä lyhyitä videoita. Teknistä tukea on mahdollista saada ainoastaan viralliselta keskustelupalstalta tai sähköpostin välityksellä. Toistaiseksi suljetun

lähdekoodin projektista löytyy myös useita virheitä, joita kehittäjät ovat raportoineet keskustelupalstalla.

NinevehGL ei ole varsinainen pelimoottori, joten pelinkehityksen näkökulmasta Unity ja Cocos2d ovat ominaisuuksiltaan paljon sen edellä. Se ei tarjoa minkäänlaista graafista käyttöliittymää, integroitua fysiikan laskentaa, reaaliaikaisia varjoja tai partikkelien luomista. Se ei ole myöskään järjestelmäriippumaton, vaan vaatii Mac OS X- tai iOS-käyttöjärjestelmän toimiakseen. Kirjastoa kehittävä DB-Interactively on julkaissut sivuillaan luettelon seuraavien versioiden mukana tulevista ominaisuuksista, jotka tulevat paikkaamaan edellä mainittuja puutteita [21]. Matalasta kypsyysasteesta huolimatta NinevehGL valittiin yhdeksi insinöörityön pelisovelluksen pääteknologioista. Testit osoittivat sen täyttävän projektin tekniset vaatimukset. Suurimpana riskinä pidettiin kirjastossa piileviä tuntemattomia kriittisiä virheitä tai ongelmia, joiden korjaaminen olisi käytännössä mahdotonta ilman kirjaston kehittäjien avustusta. Pahimmassa tapauksessa Apple voi kieltäytyä julkaisemasta epäluotettavasti toimivaa ja kaatuilevaa sovellusta App Storessa.

4 Bounceroids-pelisovellus

4.1 Konsepti

Insinöörityönä tehdyssä Bounceroids-pelisovelluksessa pelaajan tehtävänä on suojella pelialueen keskellä sijaitsevaa planeettaa sitä uhkaavilta asteroideilta. Asteroidit lähestyvät planeettaa Tower Defence -lajityypin tyyppisissä aalloissa satunnaisista suunnista pelialueen ulkopuolelta [23]. Pelaaja kontrolloi planeettaa ympäröivää suoja-aluetta, johon osuessaan asteroidi kimpoaa takaisin avaruuteen. Pelin edetessä asteroidien määrä ja nopeus kasvavat vaikeuttaen planeetan suojaamista. Peliä ei voi päästä lävitse, vaan pelaajan tavoite on kerätä pisteitä ja selviytyä mahdollisimman pitkään elossa.

Pelimekaniikka ja kontrollit suunniteltiin helposti opittaviksi. Peli ei sisällä minkäänlaista kirjallista tai visuaalista tarinankerrontaa. Kohdeyleisö ovat lapset ja nuoremmat pelaajat, jotka haluavat pelata yksinkertaisia, nopeatempoisia Arcade-lajityypin pelejä pieninä annoksina esimerkiksi koulu- tai työmatkalla.

4.2 Tekniset vaatimukset

Pelin konseptin mukainen pelikokemus vaatii suuren näytön, jonka kuvasuhde on mahdollisimman neliömäinen, jotta pelaaja näkee ruudun keskellä sijaitsevaa planeettaa lähestyvät asteroidit kaikista suunnista. Tämän vaatimuksen myötä iPhone-älypuhelinten 3:2-kuvasuhdetta ja alle neljän tuuman näyttöjä ei pidetty hyvänä vaihtoehtona kohdelaitteeksi. Sen sijaan iPad-taulutietokoneiden 4:3-kuvasuhteen ja lähes kymmenen tuuman näytön todettiin täyttävän vaatimukset älypuhelimia paremmin. Sen lisäksi taulutietokoneiden tehokkaampien prosessoreiden ja näytönohjaimien uskottiin antavan enemmän vapauksia teknisten ratkaisujen suunnittelussa.

Pelin sisäinen maailma on kolmiulotteinen, ja 3D-grafiikan piirtämistä helpottamaan valittiin ilmainen NinevehGL-kirjasto. NinevehGL antaa ohjelmoijalle mahdollisuuden hyödyntää useita eri säikeitä, joten kohdelaitteen minimilaitteistovaatimukseksi asetettiin moniydinprosessori. iPad 2 on vanhin iOS-käyttöjärjestelmän laite, joka täyttää tämän vaatimuksen. Käyttöjärjestelmän minimiversioksi asetettiin iOS 6.0, mikä varmistaa samalla, ettei peliä ole mahdollista pelata ensimmäisen sukupolven iPadilla. Ensimmäisen sukupolven iPad ei testien perusteella ollut tarpeeksi tehokas laite tavoitellun pelikokemuksen saavuttamiseksi.

iPad-taulutietokoneiden näyttöjen virkistystaajuus on 60 Hz, joka on samalla pelin maksimi kuvataajuus [24]. Kuvataajuus kertoo, kuinka monta kuvaa näytölle piirretään yhden sekunnin aikana. Mitä korkeampi kuvataajuus, sitä sulavammalta liikkeet näyttävät kuvassa. Videopeleissä kuvataajuudelle ei ole tarkkaa standardia, ja se vaihtelee tyypillisesti välillä 30–60 Hz, kun taas elokuvissa yleisesti käytössä oleva kuvataajuuden standardi on 24 Hz [25]. NinevehGL-kirjasto mahdollistaa nelinkertaisen FSAA-suotimen käyttämisen 3D-mallien reunojen sahalaitaisuuden poistamiseksi. Sahalaitaisuuden poistaminen on hyvin työläs operaatio, ja ensimmäisessä FSAA-testissä huomattiin, että pelin kuvataajuus laski testilaitteena käytetyllä iPad 2:lla noin 30 kuvaan sekunnissa. Koska pelin etenemisnopeus sidotaan piirtorutiinien nopeuteen, on tärkeää asettaa maksimikuvataajuus, jonka avulla saavutetaan tasainen pelattavuus tilanteesta riippumatta. NinevehGL-kirjaston kehittäjät suosittelevat maksimikuvataajuuden laskemista, jos sovellus käyttää FSAA-suodinta. Suositusten takia ja vaihtelevan kuvataajuuden välttämiseksi maksimikuvataajuudeksi asetettiin 25 kuvaa sekunnissa. Alenta-

malla kuvataajuutta ja vähentämällä näytönohjaimelle suunnattujen prosessien määrää pyrittiin myös pienentämään laitteen sähkönkulutusta.

4.3 Käyttöliittymäsuunnittelu ja näkymät

Pelin käyttöliittymän ja näkymien suunnittelussa pyrittiin noudattamaan KISS-periaatetta, joka korostaa mahdollisimman yksinkertaisia ratkaisuja. Käyttöliittymien interaktiivisten elementtien toteutuksessa käytettiin iPad-käyttäjille tuttuja Applen Cocoa Touch -ohjelmistokehyksen sisältämiä valmiita käyttöliittymäkomponentteja. Käyttöliittymän väreiksi valittiin pelin avaruusaiheisen teeman vuoksi tummat siniset sävyt taustalle ja kirkkaan keltainen vastaväri interaktiivisille osille.

Splash-näkymän (kuva 4) tarkoitus on esittää käyttäjälle pelin käynnistyksen yhteydessä sovelluksen julkaisijan logo mustaa taustaa vasten. Näkymä näytetään aina, kun sovellus avataan. Logo näytetään välähtävällä animaatiolla ruudun keskelle asemoituina, minkä jälkeen käyttäjälle näytetään ruudun alareunaan keskitetty kuva, joka kertoo sovelluksen käynnistysvaiheessa tehtävien latausprosessien tilanteesta. Käynnistyksen jälkeen näkymä haihtuu pois ja pelin päävalikko ilmestyy sen tilalle.

Kuva 4. Bounceroids-pelin Splash-näkymä.

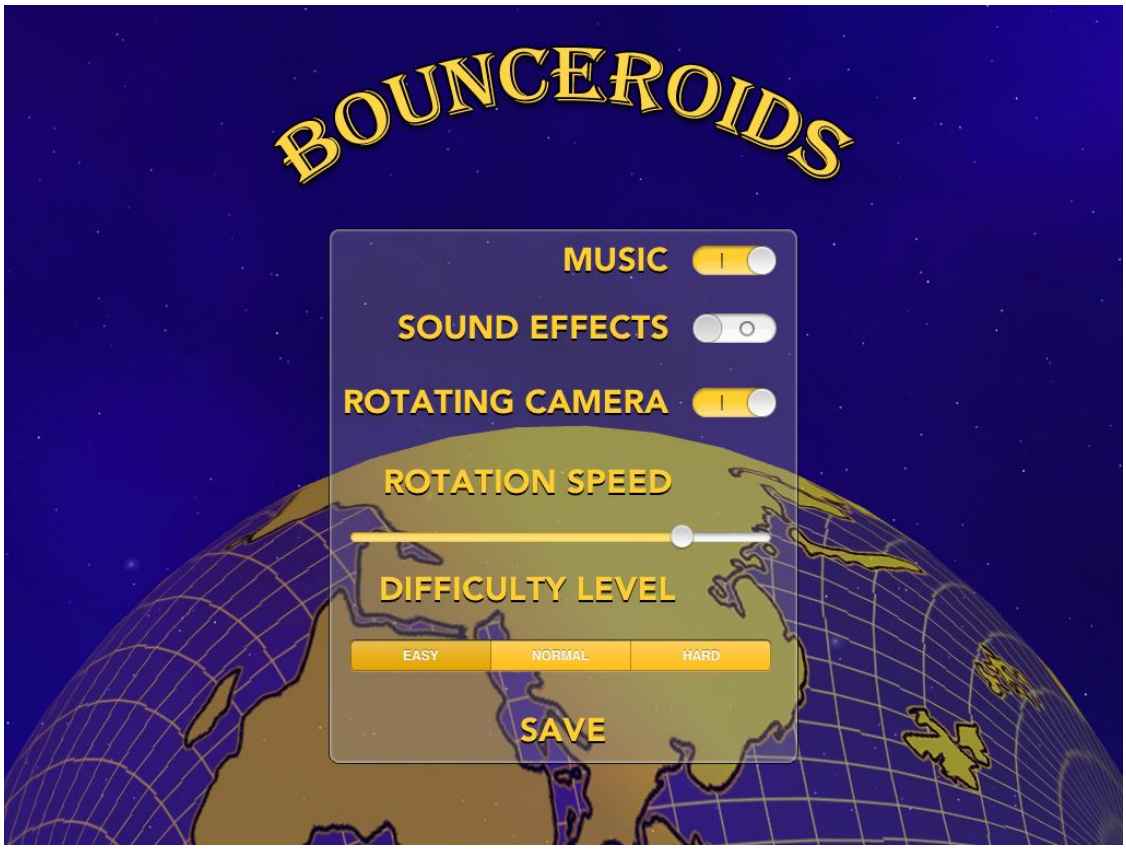
Pelin päävalikon (kuva 5) ensisijainen tehtävänä on toimia sovelluksen navigaatiorakenteen juurena, josta käyttäjä pystyy liikkumaan näkymien välillä. Toinen päävalikon tehtävistä on antaa pelaajalle selkeä käsitys pelin sisällöstä ja teemasta. Valikko käyttää taustana samaa avaruusaiheista kuvaa ja kolmiulotteista planeettaa, jotka ovat myös varsinaisessa pelitilanteessa. Samojen käyttöliittymäelementtien käytön tarkoitus on luoda jatkumoa eri näkymien välille. Päävalikossa suuri planeetta on keskitetty näytön alareunaan antamaan pelaajalle maisema ”kotiplaneetalleen”. Planeetta ja taustalla näkyvät tähdet pyörivät hitaasti akselinsa ympäri. Valikon otsakkeet on asemoitu ruudun keskellä sijaitsevaan listaan. Kun käyttäjä painaa päävalikon otsaketta, päävalikon sisältö katoaa ja sen tilalle ilmestyy valittuun otsakkeeseen liittyvä alavalikko. Päävalikko liikkuu animaatiolla vasemmalle, ja sen tilalle liikkuva alavalikko siirtyy oikealta näytön keskelle. Päävalikon otsakkeet ovat play, settings ja info.

Kuva 5. Bounceroids-pelin päävalikko.

Kuvassa 6 on Play-valikko, joka on päävalikon ensimmäinen alavalikko. Valikossa pelaaja voi aloittaa uuden pelin tai jatkaa tallennettua pelitilannetta. Valikon alareunassa on listattu pelaajan paras pistesaldo. Valikko sisältää kolme interaktiivista osaa, joista continue-nappula on piilotettuna, jos pelaaja ei ole pelannut peliä aikaisemmin. Back-nappula vie pelaajan takaisin pelin päävalikkoon. Play-valikon otsakkeet ovat new game, continue ja back.

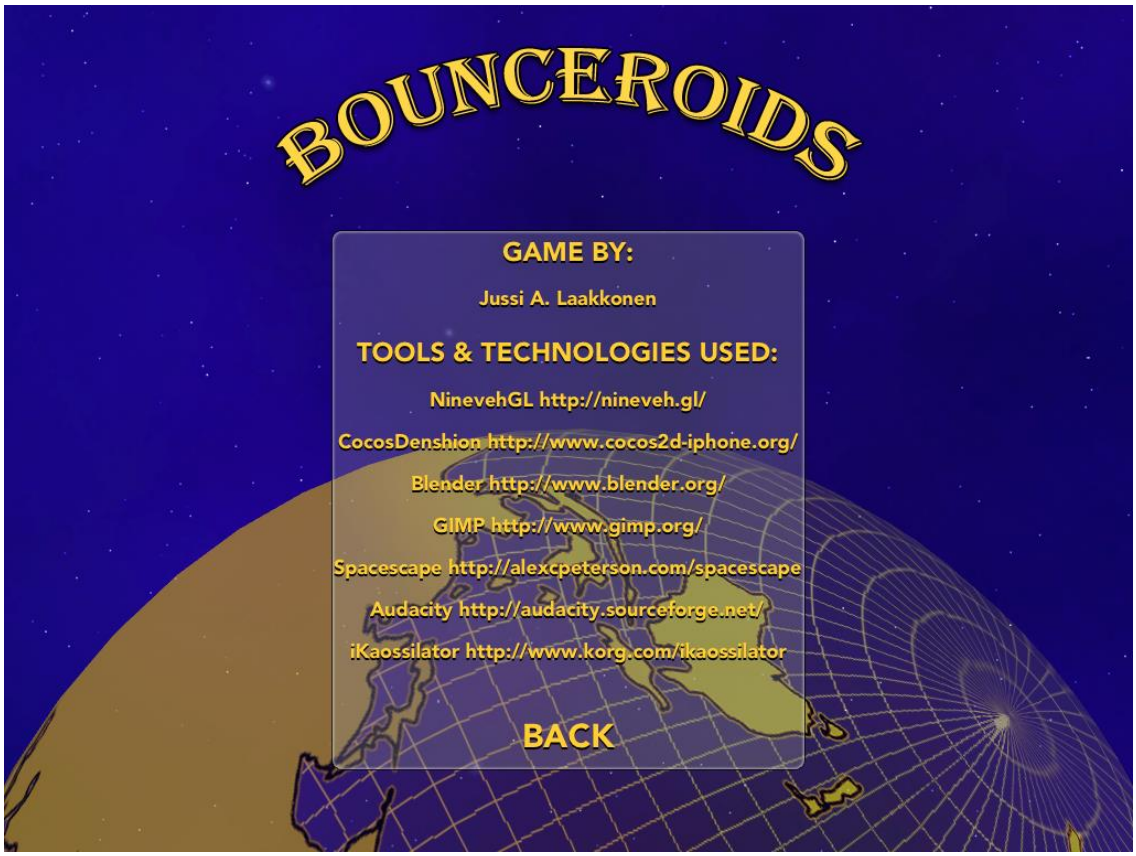
Kuva 6. Play-valikko.

Kuvan 7 Settings-valikko on päävalikon toinen alavalikko. Sen tarkoitus on tarjota käyttäjälle lista pelikokemukseen ja sovelluksen toimintaan vaikuttavista muokattavissa olevista asetuksista. Asetukset on asemoitu allekkain listaan näytön keskelle. Listan asetuksia muutetaan otsakkeesta riippuen eri tavoin. Esimerkiksi kameran pyörimisnopeutta kuvaavaa asetusta säädetään portaattomasti, kun taas kameran pyörimisen mahdollistava asetus on joko päällä tai pois päältä. Asetukset tallennetaan valikon alareunassa olevalla save-nappulalla, joka vie käyttäjän takaisin sovelluksen päävalikkoon. Settings-valikon asetuksia ovat music, sound effects, rotating camera, rotation speed ja difficulty level.



Kuva 7. Settings-valikko.

Info-valikko on päävalikon viimeinen alavalikko, ja se on esitetty kuvassa 8. Sen tarkoitus on antaa käyttäjälle lisätietoja sovelluksesta, tekijöistä ja sovelluksen sisältämistä teknologioista. Valikon yläreunassa on sovelluksen kehittäjän nimi. Nimen alapuolelle on listattu sovelluksen kehityksessä käytettyjä teknologioita. Listan otsaketta painamalla sovellus sulkeutuu ja käyttäjälle aukeaa otsakkeeseen liittyvän teknologian verkkosivusto. Valikko suljetaan alareunaan asemoidulla back-nappulalla, joka vie käyttäjän takaisin sovelluksen päävalikkoon.



Kuva 8. Info-valikko.

Singleplayer-näkymä (kuva 9) on pelikokemuksen kannalta olennaisin näkymä, koska varsinainen pelaaminen tapahtuu sen kautta. Yksinkertaisen pelimekaniikan ansiosta näkymä ei vaadi suurta ja monimutkaista käyttöliittymää, vaan se pyrittiin pitämään pelkistettynä. Planeetta ja pelaajan kontrolloima suoja-alue on aseteltu näytön keskelle, jotta pelaaja näkee lähestyvät asteroidit mahdollisimman kaukaa ja pystyy reagoimaan niihin liikuttamalla suoja-alueen oikeaan kulmaan. Ennen asteroidien lähestymistä planeetan keskelle ilmestyy asteroidiaallon järjestysnumero, joka muuttuu start-nappulaksi. Nappulaa painamalla pelaaja käynnistää asteroidien lähestymisen. Jos pelaaja epäonnistuu planeetan suojaamisessa, näkymän keskelle avautuu game over -valikko. Game over -valikosta pelaaja voi siirtyä takaisin pelin päävalikkoon tai yrittää jatkaa kesken jääneestä aallosta. Pelaajan pisteet näytetään näytön vasemmassa yläreunassa. Pisteiden alapuolelle on asemoitu planeetan kestävyyttä kuvaava numero. Pelaaja kontrolloi suoja-aluetta näytön vasempaan ja oikeaan alakulmaan asemoiduilla nappuloilla. Oikealla oleva näppäin liikuttaa suoja-aluetta myötäpäivään ja vasen näppäin vastapäivään. Nuolinappuloiden lisäksi ainoa käyttöliittymän esillä oleva interaktiivinen osa on oikeassa yläkulmassa sijaitseva pause-nappula, joka pysäyttää pelin ja

avaa pause-valikon, josta käyttäjä voi tallentaa pelitilanteen ja siirtyä takaisin päävalikkoon.

Kuva 9. Singleplayer-näkymä.

4.4 Tekninen toteutus

4.4.1 Kehitysympäristö

Pelinkehityksen päätyökaluna toimi integroitu kehitysympäristö Xcode, jota on pystynyt käyttämään versiosta 3.1 lähtien iOS-sovellusten luomiseen [26]. Applen Cocoa Touch -ohjelmistokehys sisältää tarvittavat käyttöjärjestelmärajapinnat pääasiassa Objective-C-ohjelmointikielellä, joka on C-kielen päälle rakennettu olio-ohjelmoinnin mahdollistava laajennus. Kielen syntaksi on ottanut vaikutteita Smalltalk-ohjelmointikielestä, ja se eroaa huomattavasti muista merkittävistä ohjelmointikielistä, kuten Java, C++ tai PHP.

C-tyyppiseen syntaksiin tottuneet ohjelmoijat pitävät kieltä usein outona ja vaikeaselkoisena. Eroavaisuuksien havainnollistamiseksi koodiesimerkissä 1 on klassinen Hello World -sovellus toteutettuna Objective-C- ja Java-ohjelmointikielillä. Objective-C on tällä hetkellä Applen ylläpitämä ohjelmointikieli ja suurin osa sen käytöstä tapahtuu Mac OS X- ja iOS-käyttöjärjestelmissä. Mac OS X -käyttöjärjestelmässä kielen kääntämiseen käytetään tyypillisesti Apple LLVM -kääntäjää. Muissa käyttöjärjestelmissä kielen kääntämiseen voidaan käyttää GCC-kääntäjää, joka on Unix-pohjaisten käyttöjärjestelmien yleisin ja monipuolisin kääntäjä. Ohjelmointia on mahdollista tehdä myös muilla kielillä, kuten C tai C++, joita käyttämällä pystytään kirjoittamaan alustariippumatonta koodia. Toistaiseksi kaikissa iOS-käyttöjärjestelmän laitteissa on ARM-pohjainen prosessoriarkkitehtuuri, joten koodi täytyy kääntää ARMv7- tai ARMv7s-yhteensopivaksi.

```
//Objective-C “Hello World”
@interface HelloWorld : NSObject  //Luokan esittely
-(void)hello
@end

@implementation HelloWorld
-(void)hello
   NSLog(@"Hello, World!");
}
@end

int main(int argc, char *argv[]) {
    @autoreleasepool {;
        [[HelloWorld new] hello];  //metodikutsu Objective-C:ssä
    }
    return 0;
}

//Java “Hello World”
public class HelloWorld {  //Luokan esittely
   public static void main(String[] args) {
       this.hello();    //metodikutsu Javassa
   }
 public void hello() {
     System.out.println("Hello, world!");
   }
}
```

Koodiesimerkki 1. Objective-C- ja Java-ohjelmointikielillä toteutettu Hello World -sovellus.

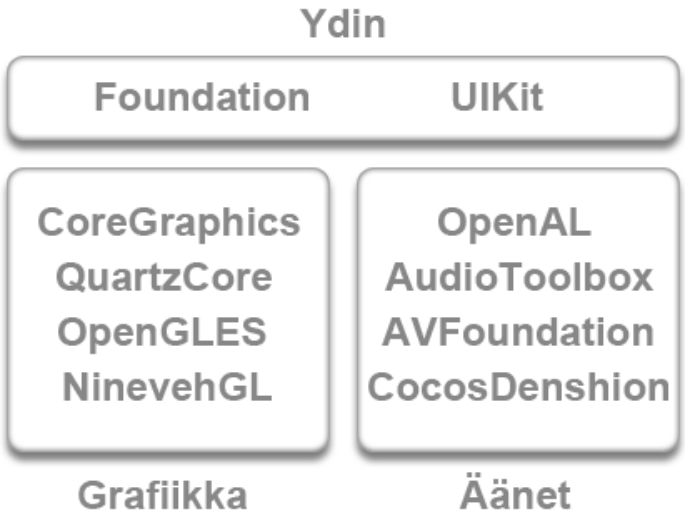
Sovelluksiin suunniteltavat käyttöliittymät voidaan rakentaa joko kirjoittamalla koodia tai graafisen käyttöliittymän avulla Xcode-ympäristöön integroidulla Interface Builder -työkalulla, joka muodostaa käyttöliittymistä xml-tyyppisen tiedoston. Interface Builder sisältää suuren määrän valmiita käyttöliittymäkomponentteja, joista osa on vapaasti muokattavissa. Apple tarjoaa ohjelmistokehittäjille iOS UI Element Usage Guidelines -dokumentin, joka sisältää ohjeita ja suositeltavia tapoja käyttöliittymien luomiseen [27]. iOS-sovellusta kehitettäessä tulee muistaa, että Apple kieltää käyttöjärjestelmän tar-

joaman Cocoa Touch -ohjelmistokehyksen yksityisten rajapintojen käytön ja saattaa kieltää sovelluksen julkaisun, mikäli se rikkoo käyttöliittymille asetettuja vaatimuksia.

Insinöörityön versionhallintaohjelmistoksi valittiin Linus Torvaldsin luoma Git. Alun perin Linux-käyttöjärjestelmän ytimen kehittämiseen tarkoitettu Git mahdollistaa hajautetun versionhallinnan ja paikalliset kopiot, jotka sisältävät kaikkien tiedostojen versiohistorian. Paikallinen kopio toimii samalla varmuuskopioina siltä varalta, että ulkopuolisella palvelimella säilytettävät tiedot katoavat tai tuhoutuvat. Git kannustaa käyttäjiä tekemään ohjelmistokehitystä omissa kehityshaaroissa, jotka liitetään myöhemmin ohjelmiston varsinaiseen päähaaraan, kun koodin on todettu toimivan oikein. Toinen varteenotettava versionhallintaohjelmisto on Subversion, jonka todettiin olevan Git-ohjelmistoa huonompi vaihtoehto sen keskitetyn luonteen takia. [28.]

4.4.2 Riippuvuudet

Pelin teknisessä toteutuksessa pyrittiin käyttämään vain ilmaisia ja parhaassa tapauksessa avoimen lähdekoodin teknologioita, koska peli pyrittiin toteuttamaan mahdollisimman pienellä budjetilla. Sovelluksen pohjana toimi NinevehGL-kirjaston Single View -kehys, joka sisältää Hello World -tyyppisen sovelluksen, jossa näytölle piirtyy pyörivä 3D-kuutio. Kehyksen käyttö säästää aikaa, koska projektiin ei tarvitse lisätä manuaalisesti NinevehGL:n vaatimia ohjelmistokehyksiä. NinevehGL:n lisäksi ainoa kolmannen osapuolen ohjelmistokehys, joka lisättiin mukaan on Cocos2d:n äänikirjasto CocosDenshion, jota käytetään pelin äänitiedostojen soittamisessa. Sovelluksen käyttämät ohjelmistokehykset voidaan jakaa kolmeen tyyppiin niiden käyttötarkoituksen mukaan. Ohjelmistokehykset ja niiden tyypit on havainnollistettu kuvaan 10.



Kuva 10. Bounceroids-pelissä käytetyt ohjelmistokehykset ja kirjastot.

Sovelluksessa käytetyistä ohjelmistokehyksistä NinevehGL ja CocosDenshion eivät tue ARC-muistinhallintaa. Kokemukseni on osoittanut, että Xcoden tarjoama ARC-konvertointityökalu ei pysty muuttamaan kaikkea koodia onnistuneesti. Työkalun käyttämisen jälkeen koodi täytyy usein käydä läpi ja korjata käsin. En päättänyt muuttaa koodia ARC-yhteensopivaksi, koska käytössäni olevat kirjastot sisälsivät suuren määrän muitten ohjelmistokehittäjien kirjoittamaa koodia, jonka pelkäsin hajoavan ARC-käännöksessä. Tästä syystä muistinhallinta jouduttiin toteuttamaan manuaalisesti. Huolimattomasti tai väärin toteutettu manuaalinen muistinhallinta voi johtaa pahimmassa tapauksessa muistivuotoihin ja sovelluksen epäluotettavaan käyttäytymiseen. Manuaalisen muistinhallinnan helpottamiseksi käytin koodiesimerkissä 2 havainnollistettua tapaa olioiden elinkaaren hallintaan.

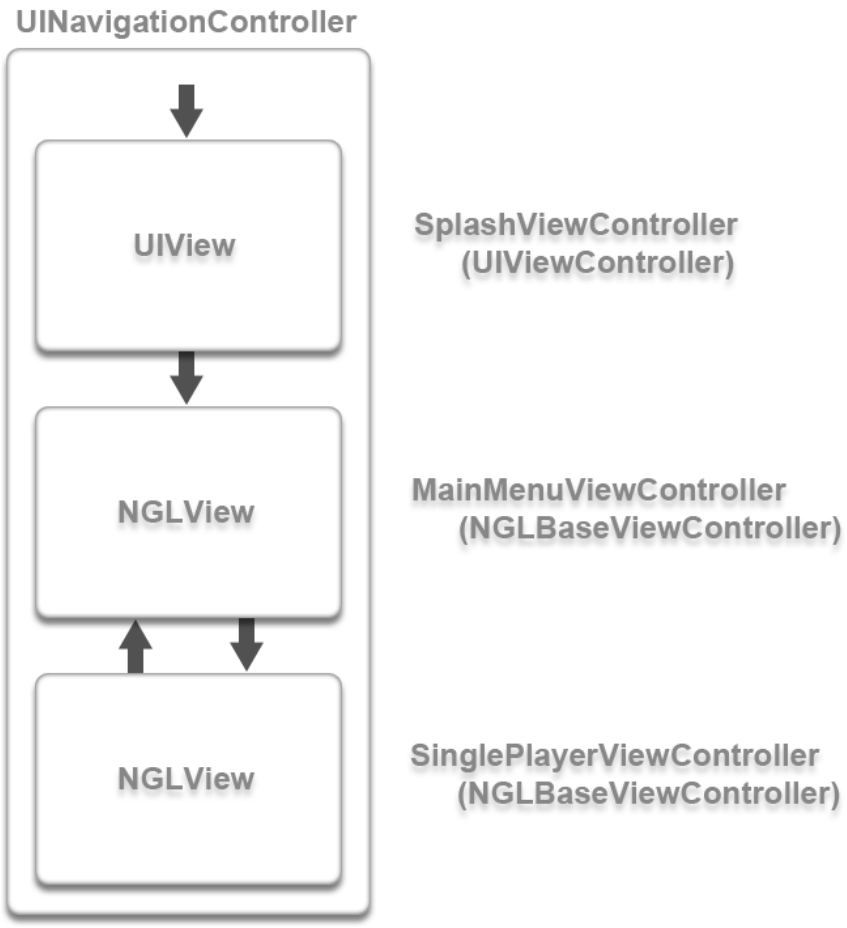
```
/*
Objective-C Muistinhallinta esimerkki. Kaikille luokan propertyille kutsutaan
automaattisesti retain-metodia, joka varmistaa, että olio on olemassa koko luokan
elinkaaren ajan. Muuttujiin asetetaan luokan sisällä ainoastaan autoreleasetut-oliot käyttäen
aina self notaatiota. Propertyihin säilötyt oliot vapautetaan luokan dealloc metodissa.
*/
@interface EsimerkkiKaksi : NSObject
@property (nonatomic,retain) NSString* tekstiYksi;
@property (nonatomic,retain) NSMutableArray *arrayYksi;
@end

@implementation HelloWorld
@synthesize tekstiYksi, arrayYksi;
-(void)esimerkkiMetodi {
   self.tekstiYksi = [NSString stringWithString: @"minun teksti!"];
  self.arrayYksi = [NSMutableArray array];
}

-(void)dealloc {
  [tekstiYksi release];
  [arrayYksi release];
  [super dealloc];
}
@end
```

Koodiesimerkki 2. Manuaalinen muistinhallinta.

4.4.3 Arkkitehtuuri

Pelin arkkitehtuurin suunnittelussa hyödynnettiin MVC (model view controller) -suunnittelumallia, joka erottelee näkymien ulkoasun, tietomallit ja tiedon käsittelijät. MVC-malli on vahvasti läsnä Applen omassa Cocoa Touch -ohjelmistokehyksen rakenteessa, ja Apple suosittelee MVC:n käyttöä iOS-sovelluksien kokonaisrakenteen suunnittelussa. Toinen pelissä runsaasti käytössä oleva suunnittelumalli on Delegation, jota käytetään muuttuvan tiedon välittämiseen eri luokkien välillä. Delegation-suunnittelumallissa delegaatti-luokka kuuntelee toisen olion tilaa ja reagoi oliossa tapahtuviin muutoksiin. Sovelluksen rakenteen pohjalla on kolme UIKit-ohjelmistokehyksen UIViewController-luokan perivää näkymää, joilla on kullakin oma

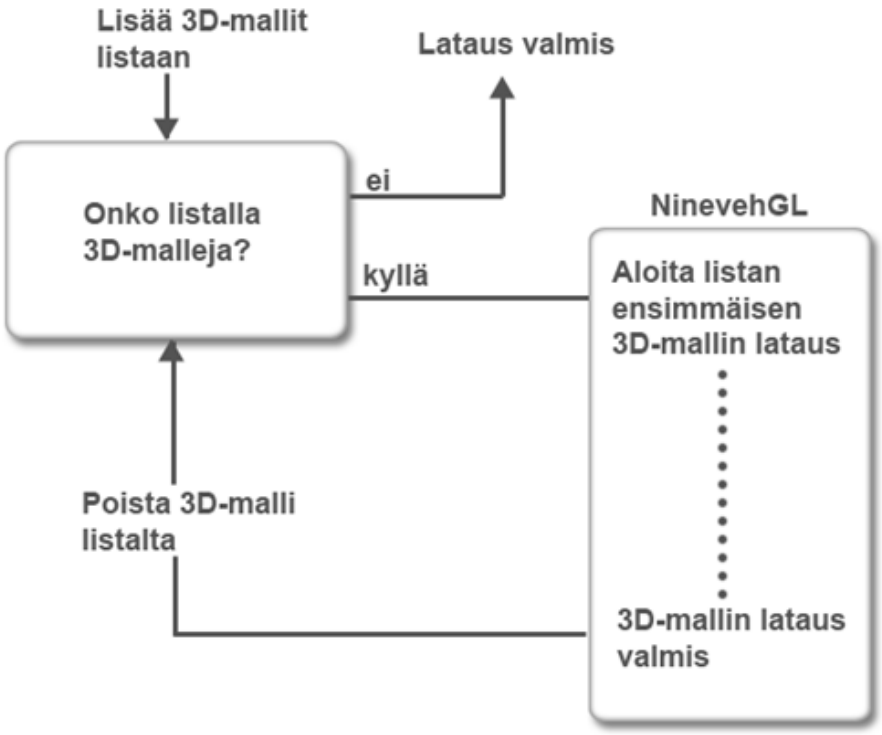
rajattu käyttötarkoituksensa ja ulkoasunsa. Näkymät on upotettu sovelluksen navigaatiorakenteen juurena toimivan UINavigationController-luokan sisälle. UINavigationControllerin käyttö mahdollistaa näkymien väliset siirtymät ilman manuaalista animointia ja näkymärakenteen hallintaa. Näkymien käyttöliittymät toteutettiin storyboard-tiedostoon. Storyboard on ollut iOS 5.0 -versiosta lähtien suositeltava tapa toteuttaa sovellusten käyttöliittymiä. Toisin kuin vanhassa lukuisiin xib-tiedostoihin perustuvassa toteutustavassa, storyboard-tyyppisessä toteutuksessa kaikki näkymät luodaan yhteen tiedostoon. Yksi storyboardin suurimmista eduista on sovelluksen rakenteen ja siirtymien selkeämpi havainnollistaminen. Kuvassa 11 on sovelluksen näkymien hierarkia ja niiden välillä tehtävät siirtymät.



Kuva 11. Bounceroids-pelin näkymähierarkia.

Näkymille, jotka sisältävät 3D-grafiikkaa, toteutettiin NGLBaseViewController-perusluokka, jonka päätarkoitus on piilottaa sisältönsä, kunnes kaikki tarvittavat 3D-mallit on ladattu näkymään. NinevehGL piirtää myös osittain ladatut 3D-mallit, eikä

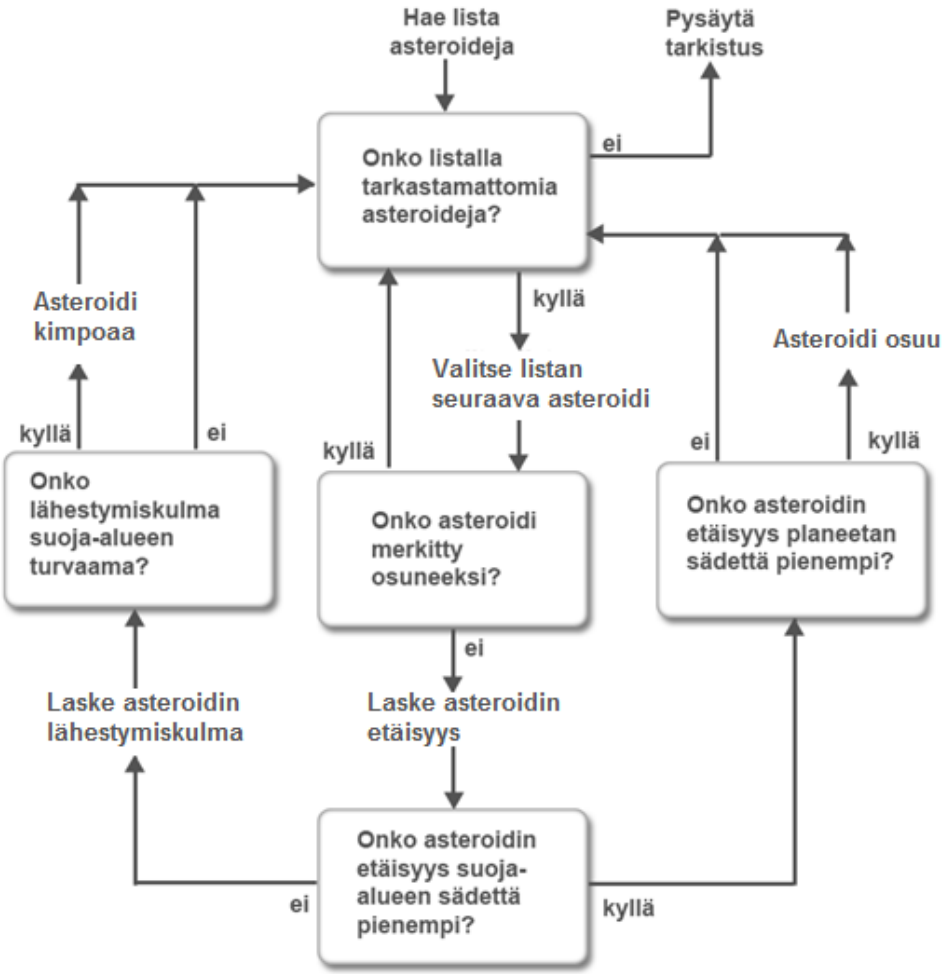
keskeneräisiä malleja haluttu näyttää käyttäjille. Kukin NGLBaseViewController toimii 3D-grafiikan lataamisesta ja piirtämisestä vastaavaan NGLScene-luokan instanssin delegaattina. NGLScene-instanssi aloittaa latausprosessin näkymään siirryttäessä ja lähettää delegaatilleen viestin latauksen päättyessä. Latauksen jälkeen näkymä voidaan näyttää pelaajalle. Grafiikka piirretään näkymän juurena toimivan NGLView-instanssin drawView()-funktiossa, jota kutsutaan 25 Hz:n taajuudella. Kuvassa 12 on korkean tason vuokaavio 3D-mallien asynkronisesta latausprosessista.



Kuva 12. 3D-mallien latausprosessi.

3D-grafiikan piirrosta vastaavalle NGLScene-perusluokalle toteutettiin kaksi eri alaluokkaa, jotka sisältävät omat piirtorutiininsa. Piirtorutiineja varten NGLScene-luokka sisältää julkisen run()-funktion, jonka alaluokat ylikirjoittavat. Perusluokan perivää MainMenuScene-luokkaa käytetään MainMenuViewController-näkymässä. MainMenuViewController-näkymä vastaa pelin päävalikkoa, eikä sen 3D-grafiikka ole interaktiivista. MainMenuScenen run()-funktio sisältää vain taustakuvan ja planeetan animointiin liittyvää koodia, eikä se sisällä monimutkaista logiikkaa. Jokaisella run()-funktion kutsulla taustakuva ja planeetta pyörivät paikallaan ennalta määritellyn astelu-vun verran.

SinglePlayerScene on toinen NGLScene-perusluokan perivä alaluokka. Sitä käytetään SinglePlayerViewController-näkymän 3D-grafiikan piirtämiseen ja pelimekaniikan hallintaan. SinglePlayerViewController-näkymä vastaa Singleplayer-näkymää, joka sisältää pelin interaktiivisen osion. SinglePlayerScene-luokkaan toteutettu run()-funktio sisältää kaiken koodin, jota tarvitaan käyttäjän kosketusten vastaanottamiseen, asteroidien liikuttamiseen ja muiden 3D-mallien animointiin. Suuren määrän koodia sisältävää run()-funktiota kutsutaan kuvataajuuden perusteella 25 kertaa sekunnissa, mikä tarkoittaa, että koodin täytyy olla hyvin optimoitua. Raskain run()-funktion sisältämä osuus on asteroidien törmäyksiä käsittelevä handleCollisions()-funktio. Kuvaan 13 on havainnollistettu handleCollisions()-funktion suorittama logiikka vuokaavion muodossa.



Kuva 13. handleCollisions()-funktion sisäinen logiikka.

Planeettaa lähestyvät asteroidit luodaan NSTimer-ajastimella, jota kutsutaan kolmen sekunnin välein. Asteroidi-3D-malli asetetaan näkymän ulkopuolelle satunnaiseen kohtaan ympyrän kehällä, jonka keskipisteen muodostaa pelaajan planeetta. Asteroidien nopeus kasvaa lineaarisesti pelin edetessä, ja nopeuteen lisätään satunnainen murtoluku. Asteroidien törmäyksien laskemista optimoitiin käsittelemällä vain ne asteroidit, jotka ovat tarpeeksi lähellä planeettaa. Laskutoimituksia ei myöskään tehdä asteroideille, jotka ovat aikaisemmin osuneet planeetan suoja-alueeseen. Asteroidien liikkeiden laskemista varten sovellukseen kirjoitettiin vektorimatematiikkaa sisältävä ERMath-luokka. Kun asteroidin lasketaan törmänneen planeettaan tai suoja-alueeseen, SinglePlayerScene ilmoittaa siitä delegaattinaan toimivalle SingePlayerViewController-näkymälle, joka päivittää pistelaskuria ja planeetan kestävyyttä kuvaavaa numeroa. Jos planeetan kestävyys laskee nollaan ja asteroidin törmäys tuhoaa planeetan, peli pysäytetään ja pelaajalle näytetään Game over -valikko.

4.4.4 3D-grafiikka

3D-mallien toteuttamisessa käytettiin päätyökaluna Blender-3D-mallinnusohjelmaa, joka on ilmainen avoimen lähdekoodin ohjelmisto. Blender on suunniteltu järjestelmäriippumattomaksi, ja se sisältää muokattavissa olevan graafisen käyttöliittymän ja animaatioon, mallintamiseen, renderöintiin ja pelinkehitykseen liittyviä työkaluja [29]. Se on laajassa käytössä elokuvateollisuudessa, ja osaa sen ominaisuuksista on pidetty ammattilaisten vertailussa jopa kaupallisia tuotteita parempana [30].

3D-mallit koostuvat 3D-avaruudessa sijaitsevista pisteistä, jotka yhdistetään monikulmioisiksi tasoiksi. Yksinkertaisimmat kolmiulotteiset tasot ovat kolmiot, joiden lukumäärää voidaan käyttää arvioitaessa 3D-mallin monimutkaisuutta ja sen piirtämiseen käytettävää aikaa näytönohjaimessa. Koska pelisovellukset piirtävät 3D-grafiikkaa reaaliaikaisesti, näytönohjaimen käsittelemän tiedon määrä vaikuttaa suoraan pelin kuvataajuuteen. 3D-mallien suunnittelussa tulee ottaa huomioon näytönohjaimen suorituskyky ja tavoiteltu kuvataajuus. Bounceroids-peliin luotiin useita eri 3D-malleja, jotka on listattu taulukkoon 1 mallin kolmioiden lukumäärän mukaan.

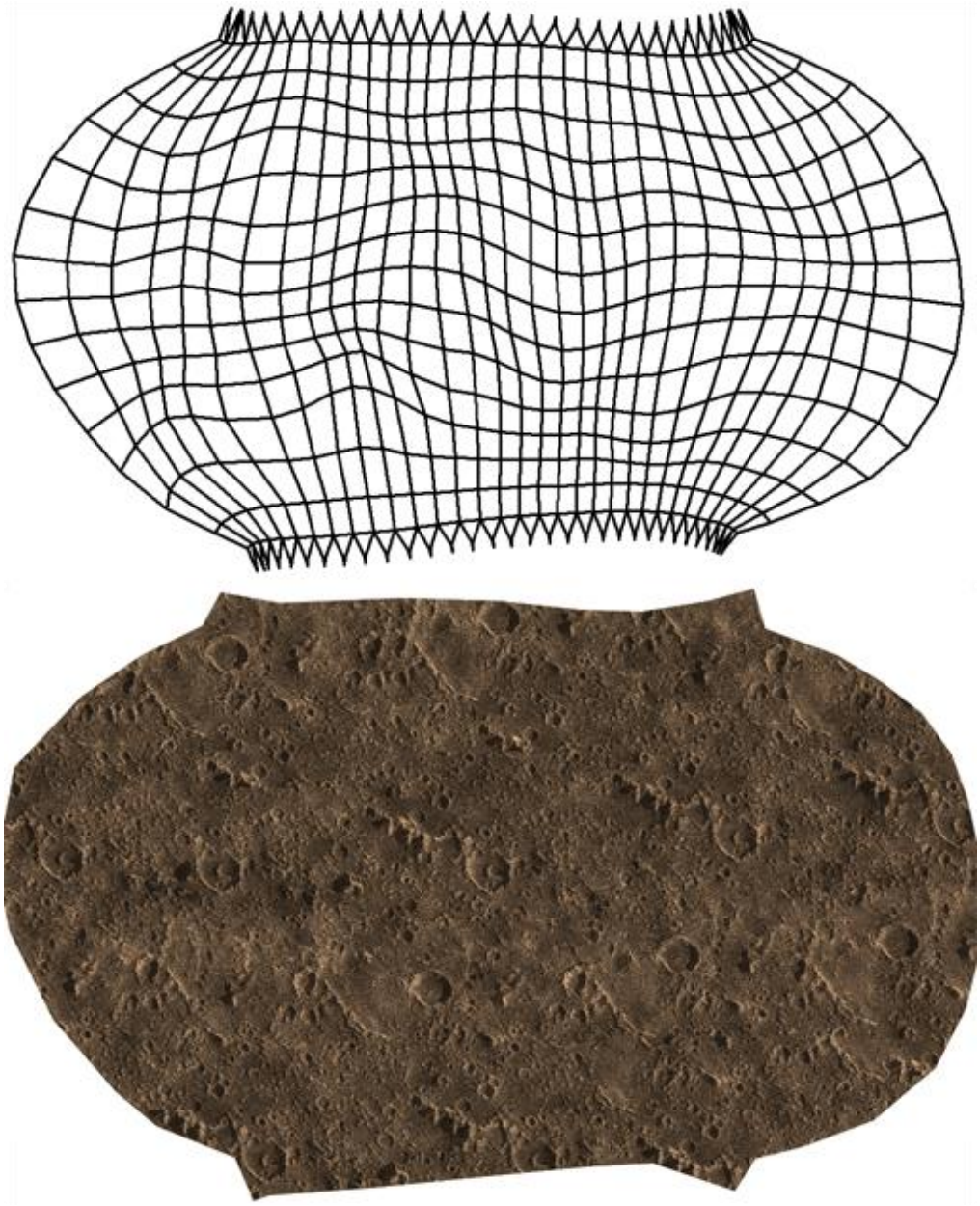
Taulukko 1. Sovellukseen luodut 3D-mallit.

Nimi	**Kolmiot**	**Tekstuurin koko**
Tausta	2	1024 x 1024
Planeetta	2600	512 x 512
Asteroidi	1088	512 x 512
Suoja-alue	88	64 x 64

Näytönohjaimen näkökulmasta monimutkaisin pelin 3D-malleista on planeetta, joka toteutettiin yksityiskohtia säästelemättä. Syy tähän oli tarve saada planeetalle tarpeeksi pyöreä muoto, jotta lähietäisyydeltä päävalikossa näytettävän mallin kulmat eivät olisi helposti havaittavissa. Muut 3D-mallit näkyvät näytöllä varsin pienessä koossa, ja ne toteutettiin vähemmillä yksityiskohdilla. Asteroidin mallinnuksessa otettiin huomioon, että asteroideja voi olla pelialueella enintään kymmenen. Tällöin Singleplayer-näkymässä piirrettävien kolmioiden suurin määrä on 13 566, joka osoittautui testien perusteella olevan tarpeeksi alhainen tasaisen 25 Hz:n kuvataajuuden saavuttamiseksi. Kaikki 3D-mallit muutettiin OBJ-tiedostomuotoon, jota NinevehGL-kirjasto osaa lukea.

3D-mallien pintojen toteutuksessa käytettiin uv mapping -tekniikkaa, jossa mallin kolmiulotteiset tasot levitetään kaksiulotteiselle alueelle, jonka päälle voidaan piirtää kuvankäsittelyohjelmalla bittikarttakuva eli tekstuuri. Planeetalle piirrettiin diffuse map- ja alpha map -tekstuurit. Muille 3D-malleille ei piirretty alpha map -tekstuuria, koska sen ei uskottu tuovan lisäarvoa mallien pienen koon ja tavoitellun visuaalisen ilmeen takia. Pelin taustalla pyörivä kuva luotiin Alex Petersonin kehittämällä avoimen lähdekoodin Spacescape-sovelluksella, joka on tarkoitettu saumattomien avaruusmaisemien luomiseen [31].

Kuvassa 14 on esitelty asteroidi-3D-mallin tasot uv-koordinaatistossa ja tasojen päälle piirretty tekstuuri. Tekstuuria luotaessa on tärkeää, että reunalle jäävien tasojen väliin ei muodostu näkyvää leikkauskohtaa. Siitä syystä kartan tasojen reunoilla sijaitsevien pikseleiden värien tulee sopia viereisten tasojen väreihin.



Kuva 14. Asteroidi-3D-mallin tasot ja tekstuuri.

4.4.5 2D-grafiikka

Pelissä esiintyvien graafisten elementtien ja 3D-mallien tekstuurien piirtämisessä käytettiin GIMP-kuvankäsittelyohjelmaa. GIMP on ilmainen avoimen lähdekoodin ohjelmisto, joka tarjoaa kaupallisten vastineiden tapaan suuren määrän ominaisuuksia ja työka-

luja ammattimaiseen kuvankäsittelyyn. Se on käyttöjärjestelmäriippumaton, ja siitä on Linux-, Windows- ja Mac OS X -versiot. [32.]

Visuaalisen ilmeen suunnittelussa käytettiin keltaista ja sinistä, jotka ovat vastavärejä. Näkymiin luodut interaktiiviset osat määriteltiin keltaisiksi ja taustat sinisiksi. Keltaisen ja sinisen värin yhdistelmä on hyvin yleisesti käytössä elokuvien ja videopelien julisteissa, joista Bounceroids-pelin värimaailma on saanut inspiraationsa. Kuvassa 15 on hyvä esimerkki keltaisen ja sinisen värin käytöstä Mass Effect 2 -videopelin julisteessa.

Kuva 15. Mass Effect 2 -videopelin juliste [33].

4.4.6 Äänet

Kaikki pelissä käytetyt äänitehosteet ja taustamusiikki luotiin iKaossilator-iPhone-sovelluksella. Se on älypuhelimille toteutettu versio Korg Kaossilator -syntetisaattorista, joka mahdollistaa melodioiden ja lyhyiden kappaleiden luomisen valmiin äänikokoelman avulla. Äänet tallennetaan laitteen kiintolevylle wav-tiedostomuodossa, minkä jälkeen ne voi siirtää tietokoneelle iExplorer-ohjelmiston kautta. Äänien leikkaamisessa käytettiin ilmaista Audacity-ohjelmistoa, minkä jälkeen lyhyet äänet muutettiin Applen suositusten mukaisesti pakkaamattomaan caf-tiedostomuotoon ja taustamusiikki pakattiin aifc-tiedostomuotoon. Apple suosittelee ima4 adpcm -algoritmin käyttöä äänten

pakkauksessa, koska iOS-käyttöjärjestelmä tukee sen purkamista ilman erillisiä kirjastoja. [34.]

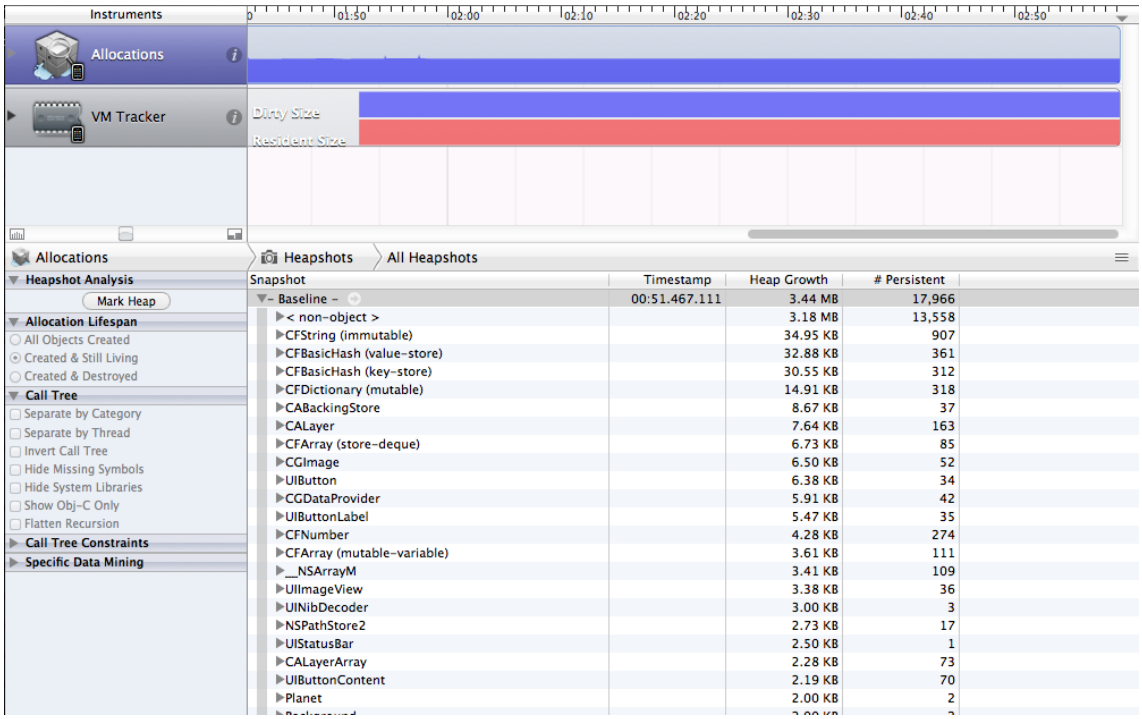
Pelin äänitehosteet suunniteltiin avaruusaiheisen teeman mukaiseksi. Pelissä käytetään korkeita ja lyhyitä ääniä pelaajan kannalta positiivisissa tilanteissa. Matalat ja voimakkaat äänet esiintyvät pelaajalle kohtalokkailla hetkillä. Äänitehosteiden lisäksi peliin luotiin kaksi eri taustamusiikkia, joista ensimmäinen soi päävalikossa ja toinen Single-player-näkymässä. Taustamusiikki on ambient-tyyppistä konemusiikkia.

4.4.7 Testaaminen

Sovellukseen toteutettiin yksikkötestejä ainoastaan pelaamisen kannalta kriittiselle ERMath-luokalle, joka vastaa 3D-avaruudessa sijaitsevien kappaleiden liike- ja törmäysvektoreiden laskemisesta. Yksikkötesteillä pyrittiin varmistamaan, että kirjoitetut matemaattiset funktiot toimivat kaikissa mahdollisissa tilanteissa.

Seuraavaksi testaamisessa käytettiin Xcoden sisältämää Instruments-työkalua (kuva 16), jolla tarkkailtiin sovelluksen muistinkäyttöä mahdollisten muistivuotojen löytämiseksi. Muistivuotoja ei löydetty. Sovelluksen todettiin käyttävän noin 4 megatavua keskusmuistia.

Sovelluksen suorituskyky testattiin iPad 2 -taulutietokoneella. Testien perusteella FSAA-suodin oli suurin suorituskykyyn vaikuttava tekijä. Retina-näytöllä varustettua iPadia ei saatu testattavaksi. Tämän vuoksi FSAA-suodin on poistettu käytöstä, jos laitteessa on retina-näyttö. Piirrettävien 3D-mallien määrällä tai yksityiskohdilla ei havaittu olevan yhtä merkittävää vaikutusta suorituskykyyn.



Kuva 16. Xcoden Instruments-työkalu.

4.5 Julkaiseminen

Peli julkaistaan App Store -sovelluskaupassa. Kauppa avattiin vuonna 2008, jolloin se sisälsi 500 sovellusta [35]. Sovellusten määrä on kasvanut nopeasti viimeisen neljän vuoden aikana, ja sovellusten välinen kilpailu on kovaa. Apple ilmoitti uuden iOS 6.1 -päivityksen yhteydessä App Storen ylittäneen 800 000 sovelluksen rajan [36]. Apple myy sovelluskehittäjille kehittäjälisenssiä, joka maksaa 99 dollaria vuodessa [37]. Kesäkuussa 2012 Applen toimitusjohtaja Tim Cook kertoi, että App Store on tuottanut sovelluskehittäjille yli 5 miljardia dollaria viimeisen neljän vuoden aikana [38].

Ennen julkaisemista sovellukselle täytyy luoda lyhyt tiivistelmä, kuvake ja ainakin yksi kuvankaappaus. Sovellus lähetetään suoraan Xcode-ympäristöstä Applen palvelimelle, minkä jälkeen se siirtyy App Storen arviointijonoon. Julkaisijoiden ylläpitämän tietokannan perusteella arviointijonon pituus on 5–10 päivää [39]. Apple voi hylätä arvioitavan sovelluksen, jos se ei läpäise laadunvalvontaa tai rikkoo App Store Review Guidelines -dokumentin ohjeita ja säännöksiä. Hylätyn sovelluksen voi lähettää arviointiin uudestaan.

Pelin julkaisijana toimii Chilipalm Games, joka on vuoden 2012 alussa perustamani iOS-pelinkehitykseen keskittyvä toiminimi. Julkaisuun liittyvä mahdollisen markkinointimateriaalin suunnittelu ja toteuttaminen jätettiin insinöörityön ulkopuolelle. Tavoitteena on lähettää sovellus App Store -arviointiin ja julkaista sovellus toukokuussa 2013.

5 Yhteenveto

Insinöörityössä luotiin alusta loppuun yhden henkilön voimin julkaisuvalmis Bounceroids-pelisovellus. Samalla toteutui pitkäaikainen henkilökohtainen haave kolmiulotteisen pelisovelluksen toteuttamisesta. Työssä käytettiin hyvin laajasti hyödyksi mediatekniikan eri osa-alueilla käytössä olevia menetelmiä ohjelmistokehityksestä, graafisesta suunnittelusta, 3D-mallinnuksesta ja äänituotannosta. Projektin laajuus ja tekniset vaatimukset onnistuttiin rajaamaan omien kykyjeni mukaisiksi, minkä takia varsinainen kehitysprosessi eteni suoraviivaisesti ilman ylitsepääsemättömiä haasteita. Odotetusti suurin osa sovelluskehitykseen käytetystä ajasta kului ohjelmointiin. Ohjelmoinnissa vaikein asia oli sovelluksen arkkitehtuurin suunnittelu ja järkevien rajapintojen luominen 3D-pelimaailman ja UIKit-pohjaisten näkymien välille. Yhteensä sovellukseen kirjoitettiin noin 3 200 riviä koodia.

Mobiilipeleistä on muodostunut lyhyessä ajassa vakavasti otettava osa peliteollisuutta. Suurimmat syyt ovat sekä päätelaitteissa että laajakaistayhteyksissä käytettävän teknologian nopea kehittyminen, hintojen lasku ja siirtyminen kokonaan digitaaliseen jakeluun. Apple ja Google ovat alentaneet kynnystä uusille pelinkehittäjille tarjoamalla valmiit jakelukanavansa pienemmillä taloudellisilla riskeillä. Kenties kuka tahansa voi luoda ja julkaista seuraavan Angry Birdsin.

Applen iOS-käyttöjärjestelmälle löytyy hyvin laaja valikoima sekä ilmaisia että kaupallisia pelinkehitykseen tarkoitettuja teknologioita. Teknologian valinnassa on ensimmäiseksi syytä ottaa huomioon pelisovelluksen tekniset vaatimukset. Mikäli jokin kaupallinen teknologia täyttää vaatimukset, on otettava huomioon tuotteen hinta ja sen käyttämisestä muodostuva taloudellinen riski. Kaupallisille teknologioille löytyy usein avoimen lähdekoodin vaihtoehto, joka ei maksa mitään. Pelinkehittäjän ei kannata pelätä uusia, kehitysvaiheessa olevia työkaluja, jotka voivat yllättää sekä hyvällä että huonolla taval-

la. Vaihtoehtoisiin teknologioihin tutustuminen antaa pelinkehittäjälle tarvittavaa perspektiiviä ja auttaa päätöksenteossa.

Ennen pelinkehityksen aloittamista suurimmaksi riskitekijäksi arvioitu NinevehGL-kirjaston kehitysversio osoittautui toimivaksi ja varsin hyväksi ilmaiseksi vaihtoehdoksi iOS-pelinkehityksessä. Kirjaston integrointi usean näkymän iOS-sovellukseen tuotti aluksi hieman ongelmia, koska 3D-maiseman piirtäminen täytyy pysäyttää ja käynnistää uudelleen näkymien välillä siirryttäessä. Toinen, pienempi haaste oli 3D-mallien lataaminen kiintolevyltä kesken pelaamisen, mikä aiheutti satunnaisia rinnakkaisuusongelmia sovelluksen säikeiden välillä. Varsinaisia toistettavissa olevia virheitä en löytänyt NinevehGL:stä. Kirjaston kehittymisen myötä se voi saada ympärilleen laajan kehittäjäyhteisön, kuten vertailun kohteena ollut Cocos2d.

Käyttämällä ilmaisia ja avoimen lähdekoodin teknologioita päästiin alkuperäisiin tavoitteisiin käytännössä ilman minkäänlaisia kehityskustannuksia. Sovelluksen julkaisemiseen vaadittava iOS Developer Program -tili jäi koko projektin ainoaksi kiinteäksi kustannukseksi.

Pelin ilmainen versio julkaistaan Applen App Storessa toukokuussa 2013. Seuraavana tavoitteenani on kehittää pelin pohjalle luotua teknologiaa ja julkaista siitä maksullinen versio, joka sisältää enemmän sisältöä ja kahden pelaajan moninpelimuodon. Myös graafista ilmettä voidaan parantaa sitä mukaa, kuin NinevehGL-kirjasto päivittyy.

Lähteet

1 Heydon, Paul. 2012. Fundraising 101 for Game Sector. Verkkodokumentti. <http://www.docstoc.com/docs/121210333/?key=YjdjNTAyNTIt&pass=Njg3Ny00ZjE1>. 24.5.2012. Luettu 10.2.2013.

2 Al-Jehani, Hesham. 2013. State of the Market: Gaming in Europe. Verkkodokumentti. <http://www.comscore.com/Insights/Presentations_and_Whitepapers/2013/Mobile_Gaming_in_Europe>. 23.1.2013. Luettu 10.2.2013.

3 Strong Demand for Smartphones and Heated Vendor Competition Characterize the Worldwide Mobile Phone Market at the End of 2012. 2013. Verkkodokumentti. International Data Corporation. <http://www.idc.com/getdoc.jsp ?containerId=prUS23916413#.UVAbHDcgg24>. 24.1.2013. Luettu 10.2.2013.

4 De Vere, Kathleen. 2012. Tracking growth: the iTunes app store vs Google Play. Verkkodokumentti. <http://www.insidemobileapps.com/2012/09/26/tracking-growth-the-itunes-app-store-vs-google-play/>. 26.9.2012. Luettu 14.4.2013.

5 Android and Apple iOS Capture a Record 92 Percent Share of Global Smartphone Shipments in Q4 2012. 2013. Verkkodokumentti. Strategy Analytics. <http://blogs.strategyanalytics.com/WSS/post/2013/01/28/Android-and-Apple-iOS-Capture-a-Record-92-Percent-Share-of-Global-Smartphone-Shipments-in-Q4-2012.aspx>. 28.1.2013. Luettu 10.2.2013.

6 Worldwide Mobile Phone Growth Expected to Drop to 1.4 % in 2012 Despite Continued Growth Of Smartphones. 2012. Verkkodokumentti. International Data Corporation. <http://www.idc.com/getdoc.jsp?containerId=prUS23818212#.UVAgVTcgg27>. 4.12.2012. Luettu 11.2.2013.

7 Tablet Shipments Soar to Record Levels During Strong Holiday Quarter. 2013. Verkkodokumentti. International Data Corporation. <http://www.idc.com/ getdoc.jsp?containerId=prUS23926713#.UVAjKzcgg24>. 31.1.2013. Luettu 11.2.2013.

8 IDC Raises Its Worldwide Tablet Forecast on Continued Strong Demand and Forthcoming New Product Launches. 2012. Verkkodokumentti. International Data Corporation. <http://www.idc.com/getdoc.jsp?containerId=prUS23696912#.UVAmfjcgg24>. 19.9.2012. Luettu 11.2.2013.

9 Flurry's Smartphone vs Tablet Perspectives. 2012. Verkkodokumentti. Mobile Ministry Magazine. <http://mobileministrymagazine.com/2012/11/06/flurrys-smartphone-vs-tablet-perspectives/>. 6.11.2012. Luettu 11.2.2013.

10 Mobile game developers. Verkkodokumentti. MobileGamesDb. <http://www.mobilegamesdb.com/developers/>. Luettu 11.2.2013.

11 Developer Economics 2012. 2012. Verkkodokumentti. Vision Mobile. <http://www.visionmobile.com/product/developer-economics-2012/#download-form>. Kesäkuu 2012. Luettu 11.2.2013.

12 Getting Started With Publishing. Verkkodokumentti. Google Inc. <http://developer.android.com/distribute/googleplay/publish/register.html>. Luettu 11.2.2013.

13 Handrahan, Matthew. 2012. Games dominate App Store revenue in 2012. Verkkodokumentti. <http://www.gamesindustry.biz/articles/2012-12-14-games-dominate-app-store-revenue-in-2012>. 14.12.2012. Luettu 11.2.2013.

14 Krzykowski, Matthaus. 2011. Admob Leads Market Share of Mobile Ad Networks on Android. Verkkodokumentti. <http://xyo.net/blog/admob-leads-market-share-of-mobile-ad-networks-on-android/>. 17.11.2011. Luettu 16.2.2013.

15 Meeker, Mary & Wu, Liang. 2012. 2012 Internet Trends. Verkkodokumentti. <http://www.kpcb.com/insights/2012-internet-trends>. 2012. Luettu 16.2.2013.

16 Mobile Video Advertising Soars on The Flurry AppCircle Network. Verkkodokumentti. Flurry. <http://www.flurry.com/pdf/Flurry_AppCircleClipsCrosses1Million MontlyCompletedViews_ShipsAndroid_Sep7-2012.pdf>. 7.9.2012. Luettu 16.2.2013.

17 Lunden, Ingrid. 2012. Free Apps Account for 89% Of All Downloads. Verkkodokumentti. TechCrunch. <http://techcrunch.com/2012/09/11/free-apps/>. 11.9.2012. Luettu 16.2.2013.

18 Fast Facts. Verkkodokumentti. Unity Technologies. <http://unity3d.com/company/public-relations/>. Luettu 17.2.2013.

19 Workflow. Verkkodokumentti. Unity Technologies. <http://unity3d.com/unity/workflow/>. Luettu 17.2.2013.

20 About. Verkkodokumentti. Cocos2d. <http://www.cocos2d-iphone.org/about>. Luettu 17.2.2013.

21 The NinevehGL's history and future. Verkkodokumentti. DB-Interactively. <http://nineveh.gl/docs/changelog/>. Luettu 22.2.2013.

22 Tutorials. Verkkodokumentti. DB-Interactively. <http://nineveh.gl/docs/tutorials/>. Luettu 22.2.2013.

23 Mitchell, Luke. 2008. Tower Defence: Bringing the genre back. Verkkodokumentti. <http://palgn.com.au/11898/tower-defense-bringing-the-genre-back/>. 22.6.2008. Luettu 14.4.2013.

24 What is the iPad 2's screen refresh rate. 2011. Verkkodokumentti. Apple Support Communities. <https://discussions.apple.com/thread/3221697?start=0&tstart=0>. 28.7.2011. Luettu 24.2.2013.

25 Haagensen, Roger. 2012. Framerate and Refresh, games and movies are not the same. Verkkodokumentti. GamaSutra. <http://www.gamasutra.com/blogs/RogerHgensen/20121105/180934/Framerate_and_Refresh_games_and_movies _are_not_the_same.php>. 5.11.2012. Luettu 24.2.2013.

26 Xcode 3.1 Release Notes. 2008. Verkkodokumentti. Apple Inc. <http://lists.apple.com/archives/xcode-users/2008/Jul/msg00270.html>. 11.7.2008. Luettu 20.3.2013.

27 iOS UI Element Usage Guidelines. Verkkodokumentti. Apple Inc. <http://developer.apple.com/library/ios/#documentation/userexperience/conceptual/mobilehig/UIElementGuidelines/UIElementGuidelines.html#//apple_ref/doc/uid/TP40006556-CH13-SW41>. Luettu 20.3.2013.

28 About. Verkkodokumentti. Git. <http://git-scm.com/about/>. Luettu 20.3.2013.

29 Features. 2012. Verkkodokumentti. Blender Foundation. <http://www.blender.org/features-gallery/features/>. Luettu 16.2.2013.

30 The making of Sintel. 2011. Verkkodokumentti. 3D World. <http://www.3dworldmag.com/2011/02/09/the-making-of-sintel/4/>. 9.2.2011. Luettu 16.2.2013.

31 Peterson, Alex. 2008. Spacescape. Verkkodokumentti. <http://alexcpeterson.com/spacescape>. Luettu 1.4.2013.

32 Feature Overview. 2013. Verkkodokumentti. The GIMP Team. <http://www.gimp.org/features/>. Luettu 1.4.2013.

33 Mass Effect 2 Wallpaper. 2009. Verkkodokumentti. BioWare. <http://www. reviewstl.com/wp-content/uploads/2010/01/mass-effect-wallpaper-banner-1024x576.jpg>. Luettu 1.4.2013.

34 Preferred Audio Formats in iOS. 2010. Verkkodokumentti. Apple Inc. <http://developer.apple.com/library/ios/#documentation/AudioVideo/Conceptual/MultimediaPG/UsingAudio/UsingAudio.html>. 1.9.2010. Luettu 16.2.2013.

35 Ricker, Thomas. 2008. Jobs: App Store launching with 500 iPhone applications, 25% free. Verkkodokumentti. Engadget. <http://www.engadget.com

/2008/07/10/jobs-app-store-launching-with-500-iphone-applications-25-free/>. 10.7.2008. Luettu 20.3.2013.

36 Apple Updates iOS to 6.1. 2013. Verkkodokumentti. Apple Inc. <http://www.apple.com/pr/library/2013/01/28Apple-Updates-iOS-to-6-1.html>. 28.1.2013. Luettu 23.3.2013.

37 iOS Developer Program. Verkkodokumentti. Apple Inc. <https://developer.apple.com/programs/ios/>. Luettu 23.3.2013.

38 Indvik, Lauren. 2012. App Store Stats: 400 Million Accounts, 650,000 Apps. Verkkodokumentti. <http://mashable.com/2012/06/11/wwdc-2012-app-store-stats/>. 11.6.2012. Luettu 23.3.2013.

39 Average App Store Review Times. Verkkodokumentti. Shiny Development. <http://reviewtimes.shinydevelopment.com/>. Luettu 23.3.2013.